(19) **KOREAN INTELLECTUAL PROPERTY OFFICE**

KOREAN PATENT ABSTRACTS

(11)Publication number: **1020050023805 A**

(43)Date of publication of application: **10.03.2005**

(21)Application number: **1020030061242**

(22)Date of filing: **02.09.2003**

(71)Applicant: **SAMSUNG ELECTRONICS CO., LTD.**

(72)Inventor: **CHOI, CHANG KI**
KIM, DEOK HO
KIM, YU HYOK
MOON, JEONG YEON

(51)Int. Cl **G06F 9/46**

(54) **METHOD FOR DYNAMICALLY ARRANGING APPLICATION PROGRAM MODULE USING XML**

(57) Abstract:

PURPOSE: A method for dynamically arranging an application program module using the XML (eXtensible Markup Language) is provided to modify a GUI(Graphic User Interface) or a UI by modifying an XML document without dividing the application program into multiple layers and recompiling each layer. CONSTITUTION: Multiple independent program modules are loaded. A common information repository for communicating data among the program modules is generated. The first module receiving a command from a user stores the data for performing the command to the common information repository(S620). The second module to perform the command reads the data from the common information repository(S660) and performs the command by using the read data(S665).

copyright KIPO 2005

Legal Status

Date of request for an examination (20030902)

Notification date of refusal decision (00000000)

Final disposal of an application (registration)

Date of final disposal of an application (20051229)

Patent registration number (1005629050000)

Date of registration (20060314)

Number of opposition against the grant of a patent ( )

Date of opposition against the grant of a patent (00000000)

Number of trial against decision to refuse ( )

Date of requesting trial against decision to refuse ( )

(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/46 (2006.01)

(45) 공고일자 2006년03월21일
(11) 등록번호 10-0562905
(24) 등록일자 2006년03월14일

(21) 출원번호 10-2003-0061242
(22) 출원일자 2003년09월02일

(65) 공개번호 10-2005-0023805
(43) 공개일자 2005년03월10일

(73) 특허권자 삼성전자주식회사
경기도 수원시 영통구 매탄동 416

(72) 발명자 김유혁
경기도성남시분당구야탑동대우아파트207-1404

김덕호
서울특별시서초구우면동코오롱아파트103동606호

최창기
경기도안양시만안구석수1동공영아파트가동104호

문경연
서울특별시서초구방배3동임광아파트1동601호

(74) 대리인 김동진

심사관 : 문형섭

(54) 어플리케이션 프로그램 모듈의 동적 배치 방법

요약

본 발명은 어플리케이션 프로그램 모듈의 동적 배치 방법에 관한 발명으로서, 본 발명의 실시를 위한 프로그램 모듈의 동적 배치 방법은 독립적으로 동작하는 다수의 프로그램 모듈을 로딩하는 제1단계와, 상기 프로그램 모듈간에 데이터 통신을 하기 위한 공동의 정보 저장소를 생성하는 제2단계와, 상기 모듈들 중에서 사용자로부터 명령을 입력받은 제1 모듈이 명령을 수행하기 위한 데이터를 공동의 정보 저장소에 기록하는 제3단계, 및 상기 모듈들 중에서 상기 명령을 수행할 수 있는 제2 모듈이 상기 정보 저장소로부터 데이터를 읽은 후, 상기 데이터를 이용하여 상기 명령을 수행하는 제4단계를 포함하는 것을 특징으로 한다.

대표도

도 2

색인어

어플리케이션 프로그램 아키텍처

명세서

도면의 간단한 설명

도 1은 본 발명의 실시에 따라 각각의 프로그램 모듈이 갖는 XML문서의 기본적인 구조를 나타내는 예시도이다.

도 2는 본 발명의 실시에 따라 구현된 어플리케이션 프로그램을 나타내는 예시도이다.

도 3은 도 2에서 도시한 어플리케이션 프로그램의 구성에 대한 설명을 하고 있는 XML문서를 나타내는 예시도이다.

도 4는 도 2에서 도시한 어플리케이션 프로그램에 포함된 임의의 모듈에 대한 구성에 대한 설명하는 XML문서를 나타내는 예시도이다.

도 5는 본 발명의 실시에 따른 어플리케이션 프로그램에서 하나의 모듈을 로딩(loading)하는 방법을 나타내는 일실시예 처리 흐름도이다.

도 6은 본 발명의 실시에 따른 각각의 모듈간의 데이터를 공유하는 방법을 나타내는 예시도이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 어플리케이션 프로그램 모듈의 동적 배치 방법에 관한 것으로서, 보다 상세하게는 사용자 인터페이스의 구성 또는 이미지를 소스 코드와는 별개의 문서에서 지정함으로써 재컴파일을 하지 않더라도 사용자 인터페이스 구성 또는 프로그램의 스킨(skin) 변경을 간편하게 수행하는 방법에 관한 것이다.

운영 체제(Operating System, 이하 'OS'라 한다)는 자신의 환경 내에서 프로그램을 개발할 수 있도록 특수한 목적의 일부 장치는 제외하고 OS가 인식하는 모든 장치를 제어 할 수 있는 어플리케이션 프로그램 인터페이스(Application Program Interface, 이하 'UI'라 한다)를 제공한다. API의 종류에는 파일을 관리해주는 기본적인 파일 I/O API, 미디어 파일을 제어하는 API, 통신 프로토콜을 이용하여 외부와 통신을 가능하게 해 주는 API 등 그 종류는 아주 많다. 특정한 OS 기반에서 어플리케이션 프로그램을 개발하는 개발자는 상기와 같은 OS가 제공하는 다양한 API의 특징과, 해당 OS가 하나의 프로그램 및 하나의 컨트롤을 인식하고 통제하는 방법에 대한 규칙을 이용하여 개발자가 원하는 최종 프로그램을 제작하는 과정을 밟게 된다.

결국 OS상에서 객체 지향 프로그램(Object Oriented Program) 기반의 사용자 인터페이스(User Interface, 이하 'UI'라고 한다) 어플리케이션 프로그램을 개발하는 개발자는 독립된 단위기능을 갖는 모듈 단위로 다양한 기능을 설계하고 모듈간 명령과 데이터의 전달 방법, 해당 OS에서 화면에 무엇인가를 디스플레이해 주는 방법, 이러한 정보를 바꾸기 위한 방법, 어떤 명령을 수신했을 때 어떤 반응을 보여줄 것인가에 대한 정의 방법 등을 조합하여 하나의 UI 어플리케이션 프로그램을 설계하고 개발하게 된다.

예를 들어, 윈도우 OS는 자신의 OS에서 사용하는 기본적인 형태의 프로그램을 구성할 수 있는 다양한 컨트롤을 제공하여 주고, 특별한 색깔과 방법을 이용해 화면에 그림을 그려주는 GDI등의 다양한 인터페이스를 제공하며 메시지 핸들링(Message Handling) 방식과 같은 각 컨트롤간 명령을 전달할 수 있는 방법을 제공한다.

UI 어플리케이션 프로그램을 할 때 사용되는 많은 정보를 리소스라 한다. 이에 대한 정보로는 버튼과 같은 컨트롤이 화면에 보이도록 하는 경우 버튼의 좌표와 위치, 또한 이 버튼의 체배열 형태와 같은 레이아웃(layout) 방식에 관한 정보가 있을 수 있으며, 이 버튼을 특정한 이미지로 구성하였다면 그 이미지의 정보 또한 리소스의 일부가 된다.

종래에 프로그램을 작성하는 경우에는 이러한 정보를 활용하여 소스 프로그램에 직접 코딩을 하였다. 물론, 일부는 프로그램의 외부에 노출시켜 두기도 하였지만 다수의 정보를 코드의 일부로서 상수화하거나 헤더(header) 파일에 기록을 하였다. 이렇게 포함된 값들을 활용하여 화면에 보이는 컨트롤들의 레이아웃(layout)을 어떻게 배치하는가 하는 등의 로직을

갖는 함수를 구성하고 여러 모듈간 인터페이스를 구성하였다. 이러한 어플리케이션 프로그램의 경우에는 사용자가 프로그램 벤더(vendor)가 제공하는 한가지 형태의 화면과 UI 구성만을 사용할 수 밖에 없다. 따라서, 기존의 구조를 수정하지 않고 새로운 요구 사양이 발생하였을 때, 해당 부분을 수정하기 위해서는 프로그램 내에 해당 부분과 관련이 있는 혹은 관련이 있을 수 있는 모든 부분을 검토하고 수정하여야 하며, 이 모든 과정을 종료한 후에 전체 프로그램을 재컴파일하여야 한다.

프로그램을 코딩할 때, 자기 자신이 할 수 없는 특정한 기능을 수행하기 위해서는 OS가 제공하는 API를 사용하거나 그 기능을 수행할 수 있는 클래스의 인스턴스를 생성하여 해당 인스턴스의 멤버 함수를 이용해야만 한다. 이러한 방법은 프로그램을 코딩할 때, 반드시 해당 API의 이름이나, 클래스의 이름을 알고 있어야만 코딩과 컴파일이 가능하다는 제약이 있다. 결국, 컴파일 전에 모든 기능의 리스트와 어떤 기능은 어디에서 얻어야 할 지가 결정되어야만 프로젝트의 진행이 가능했다. 하지만, 이러한 방법 역시 요구하는 기능이 다양하고 그 변경 가능성이 높은 경우에는 한 가지 기능을 추가/삭제/변경하기 위해서 해당 부분을 수정하고 전체 프로젝트를 재컴파일해야 한다는 단점이 있다.

윈도우의 API는 특별한 방법을 이용해서 컴파일하기 전에 그 이름을 모르게 진행할 수도 있지만 그 방법이 너무 복잡하여 기능이 다양한 경우에는 현실적인 해결책이 되지 못한다. 또한, 클래스 이름의 경우는 그 이름을 모르고는 해당 클래스의 인스턴스를 생성할 방법이 없으므로 컴파일 자체가 절대 불가능하게 된다.

UI 구성이 복잡한 UI 어플리케이션 중 워드 프로세서(word processor), 미디어 플레이어(media player)와 같이 파일을 직접 처리하는 프로그램의 경우에는 사용자의 요구도 다양하고 그만큼 그 입출력 방식 또한 매우 다양하다.

컴퓨터가 어떠한 장치를 인식하는 경우, 특별한 파일을 '더블클릭'이나 '엔터'의 형태로 실행하는 경우, A 모드로 동작중이던 프로그램이 B 모드로 전환되는 경우 등 그 입출력 시나리오는 아주 다양하며, 이 때 도출될 수 있는 UI 구성 역시 아주 다양하다. 기존의 프로그램에서는 프로젝트를 진행하는 프로세스 상에서 요구 기능이 도출되고 그 기능에 맞게 자신이 맡은 모듈을 구성하였고 이러한 모듈은 이 모듈의 근거가 되었던 요구 기능에만 아주 충실하게 제작이 되었다. 다양한 입출력 구성에 따라 해당 모듈의 초기화 단계에서 필요한 정보와 동작이 프로젝트 중/후반에 새롭게 변경, 추가, 삭제, 수정되며 이러한 변경이 있을 때마다 프로그램을 재수정 해야만 했다. 그 결과, 미처 예견하지 못한 다양한 문제들이 발생했고 이는 결국 버그(bug) 또는 기능 삭제를 통해 해결하였다.

따라서, UI 또는 그래픽 사용자 인터페이스(Graphic User Interface, 이하 'GUI'라 한다)를 변경하는 경우 해당 부분의 소스 코드를 변경하여 재컴파일해야 하는 불편함을 줄이고, 새로운 소프트웨어 아키텍처를 도입하여 간편하게 UI 또는 GUI를 변경할 필요성이 생기게 되었다.

발명이 이루고자 하는 기술적 과제

본 발명은 상기한 문제점을 개선하기 위해 안출된 것으로, 본 발명에서는 범용적으로 사용되고 수정이 간편한 XML을 이용하여 어플리케이션 프로그램을 다수의 계층으로 나누어 재컴파일하지 않고 XML문서의 수정만으로 UI 또는 GUI를 변경할 수 있는 방법을 제시하고 있다.

발명의 구성 및 작용

상기 목적을 달성하기 위하여, 본 발명의 실시에 따른 어플리케이션 프로그램 모듈의 동적 배치 방법은 독립적으로 동작하는 다수의 프로그램 모듈을 로딩하는 제1단계와, 상기 프로그램 모듈간에 데이터 통신을 하기 위한 공동의 정보 저장소를 생성하는 제2단계와, 상기 모듈들 중에서 사용자로부터 명령을 입력받은 제1 모듈이 명령을 수행하기 위한 데이터를 공동의 정보 저장소에 기록하는 제3단계, 및 상기 모듈들 중에서 상기 명령을 수행할 수 있는 제2 모듈이 상기 정보 저장소로부터 데이터를 읽은 후, 상기 데이터를 이용하여 상기 명령을 수행하는 제4단계를 포함한다. 바람직하게는 상기 프로그램 모듈은 각각의 모듈 구성이 XML 문서로 정의되어 있는 것을 포함한다. 이 때, 상기 상기 프로그램 모듈은 사용자 인터페이스를 이용하여 사용자와 입출력 통신을 할 수 있는 컴포넌트 모듈과, 다수의 컴포넌트 모듈을 포함하는 컨테이커 모듈과, 사용자 인터페이스를 구비하지 않고 사용자의 요구에 따라 내부적으로 처리를 하는 엔진 모듈을 포함한다.

이하, 첨부된 도면을 참조하여 본 발명의 일실시예에 따른 어플리케이션 프로그램 모듈의 동적 배치 방법에 대하여 구체적으로 설명하도록 한다.

한편, 본 발명에서는 새로운 형태의 어플리케이션 프로그램 아키텍처를 제시하고 있으므로, 본 발명에 대한 실시예를 설명하기에 앞서 본 발명에 따른 용어를 정리할 필요가 있다.

1. 컴포넌트(component)

사용자로부터 입력을 받거나, 사용자 요구에 응답하여 처리한 결과를 사용자에게 보여주는 등 사용자와 통신을 할 수 있는 UI의 구성 요소를 말한다.

2. 컨테이너(container)

다양한 컴포넌트들 중에 같은 성격을 갖는 컴포넌트들 간의 집합을 의미한다. 예를 들어 '제목 표시줄', '메뉴 바', '메인 화면'의 3개의 컴포넌트가 있다고 가정하면, '제목 표시줄' 그들 밑에는 시스템 버튼, 최소화 버튼, 최대화 버튼, 종료 버튼의 4가지 컴포넌트가 있을 수 있다. 이러한 관계에서 살펴보면 '제목 표시줄'은 그 아래의 4개의 컴포넌트를 포함하는 컨테이너이라 할 수 있고, 또한 어플리케이션 프로그램 컨테이너의 컴포넌트로도 볼 수 있다. 즉 컨테이너는 다른 컨테이너의 컴포넌트가 될 수도 있고, 다른 컴포넌트를 포함하고 있을 수도 있다.

3. 엔진(engine)

컨테이너와 컴포넌트를 사용하면 어플리케이션 프로그램의 UI 부분을 모두 설명할 수 있다. 그러나 어플리케이션 프로그램이 동작하기 위해서는 UI외에도 실질적인 기능을 하는 부분이 필요하다. 즉, 각각의 컴포넌트가 사용자의 요구에 따라 처리를 해야 하는 부분을 말하는 것으로서 UI 부분이 필요가 없는데, 본 발명에서는 이러한 기능을 담당하는 부분을 엔진(engine)이라고 한다.

어플리케이션 프로그램 사용자가 어플리케이션 프로그램에 대하여 어떠한 요구를 했는지 엔진입장에서는 알 수가 없다. 엔진은 컴포넌트가 사용자의 요구를 분석한 결과를 전달받아, 자신이 할 수 있는 일을 수행하고 그 결과를 다시 컴포넌트로 전달하게 된다. 이와 같이 엔진은 어플리케이션 프로그램 내부에 감춰져 있는 기능이 된다.

도 1은 본 발명의 실시에 따라 각각의 프로그램 모듈이 갖는 XML문서의 기본적인 구조를 나타내는 예시도이다.

도 1에서 도시한 라인 번호를 참조하여 개략적인 구조 및 엘리먼트, 속성들을 설명하도록 한다.

1라인 내지 2라인은 XML문서의 헤더부분에 해당한다.

3라인의 'Layout'엘리먼트는 속성으로 'type', 'width', 'height'를 갖고 있다.

'type'은 레이아웃(layout) 리사이즈 방식의 스타일로서, 'fix', 'resize', 'flex' 3가지가 있다. 'fix'는 레이아웃(layout)이 변경되지 않고 고정되어 있는 것을 의미하고, 크기와 위치도 변하지 않는 구조를 나타낸다. 'resize'는 레이아웃(layout)이 자유롭게 변경될 수 있는데, 사용자가 크기의 변경을 할 수도 있고, 리사이즈도 지원한다. 'flex'는 레이아웃(layout)이 위치 계산 알고리즘에 의해 결정되는 형태를 의미한다. 예를 들어, 하나의 컨테이너 안에 5개의 컴포넌트가 있으면, 5개를 균일하게 수직으로 배열하거나, 원 모양으로 배열하는 등, 계산에 의한 배치를 수행한다. 이 구조는 하나의 기능을 추가 하다라도 직접 계산하지 않고, 프로그램에서 자동으로 위치를 계산하여 보여주게 된다.

'width'는 레이아웃(layout) 의 기본 폭을 나타내고, 'height'는 레이아웃(layout)의 기본 높이를 나타낸다.

4라인의 'name'은 'Layout'엘리먼트의 이름을 나타낸다.

5라인의 'dll_path'는 'Layout'엘리먼트의 'type'가 'flex'인 경우 각각의 모듈의 위치를 계산해 주는 알고리즘 함수를 갖는 dll의 경로를 나타낸다.

6라인의 'mode'엘리먼트의 'bkgnd'는 모듈 전체의 배경 이미지를 나타낸다.

7라인의 'compts'엘리먼트는 해당 모듈이 갖는 컴포넌트들의 집합을 나타내고, 'compt_count'는 해당 모듈에 속한 컴포넌트의 개수를 나타낸다.

8라인의 'compt dll_path'는 컴포넌트의 경로를 나타내고, 'compt_layout'은 컴포넌트의 레이아웃(layout) 문서 이름을 나타낸다.

9라인의 'compt_name'은 컴포넌트의 이름을 나타낸다. 상기 컴포넌트 이름은 하나의 문서에서는 고유해야 한다.

10라인의 'compt_pos'은 컴포넌트의 위치를 나타낸다. 속성으로서 'style'은 리사이징 스타일을 나타내고, 'xPos'는 X좌표, 'yPos'는 Y좌표, 'width'는 폭, 'height'는 높이를 나타낸다.

11라인의 'compt_skin'은 컴포넌트의 스킨(skin)을 나타낸다. 속성으로서 'skin_type'은 버튼, 리스트, 배경 등과 같이 스킨(skin)이 표현되는 컴포넌트의 종류를 나타낸다. '<compt_skin>'과 '</compt_skin>'사이에는 스킨(skin) 이미지의 상대경로(path)가 놓이게 된다.

컴포넌트의 스킨(skin)의 종류에 대하여 [표 1]에서 좀더 설명하고 있다.

[표 1]

컴포넌트의 스킨(skin)종류 및 필요한 이미지 수

Skin type	설명(description)	필요한 이미지 수
BACKGROUND	컨테이너의 배경에 들어가는 스킨(skin)	리사이즈 방식에 따라 1. 'WIDTH_FIXED', 'HEIGHT_FIXED', 2. 'WIDTH_FIXED' 3. 'HEIGHT_FIXED' 각 리사이즈 정보는 모두 컴포넌트에 정의된 것을 따른다.
BUTTON	오브젝트의 왼쪽이 비율에 따라 변함	1개(Default, Hover, Pressed, Disabled가 붙은 이미지)
TREE	트리컨트롤	2개(Image List, Background)
LIST	리스트컨트롤	2개(Image List, Background)
SCROLL	스크롤바	2개(Scrollbar, Background)
COMBO	콤보박스	2개(Dropdown Button, Background)
SLIDER	슬라이드바	1개
USERDEFINE	컴포넌트에 들어가는 스킨(skin) 정보 중, 특수한 목적으로 특수 컴포넌트에서만 사용하기 위한 구조	이미지의 형태, 리사이즈 방식에 따라 달라짐

스킨(skin) 구조는 크게 2가지로 나누어진다. 하나는 배경(background)이 있는 컴포넌트이고, 다른 하나는 배경(background)이 없는 컴포넌트이다.

예를 들어, 리스트컨트롤(listcontrol) 또는 트리컨트롤(treecontrol)은 배경(background)이 있지만, 버튼(button)의 경우에는 배경(background)이 없다.

스킨에는 많은 수의 이미지가 필요하다. 각각의 상태마다 다른 이미지가 들어가기 때문에, 하나의 버튼, 하나의 스크롤 바, 하나의 배경등을 표시하더라도 적게는 1가지에서 많게는 10여 종류의 이미지가 들어가게 된다. 그러한 모든 이미지를 '<compt_skin>'과 '</compt_skin>'사이에서 모두 표현하는 것은 어렵기 때문에, 본 발명에서는 2개의 이미지를 가지고 모두를 표현하게 된다.

우선, 각 모듈의 배경을 나타내는 이미지와, 모듈 자체의 이미지, 이렇게 2개를 사용하게 된다. 모듈 자체의 이미지에서 여러 장의 이미지가 필요할 경우에는 한 개의 이미지 파일에 각각의 이미지를 모아놓고, 프로그램 상에서 각 이미지를 잘라내어 사용하도록 한다.

예를 들어, 버튼(button)에 이미지를 입히기 위해서는 일반적으로 4개의 이미지가 필요하다. 즉, 기본이미지, 마우스가 올라갔을 때의 이미지, 눌렀을 때의 이미지, 디스에이블(Disable) 상태의 이미지 등이다. 각각의 이미지들을 한 개의 파일에 특정한 순서 (예컨대, Default-Hover-Select-Disable)로 연결하여 하나의 이미지 파일을 만들고, 그 이미지 파일을 버튼을 표현하는 코드에서 잘라내어 표현을 하는 것이다. 이렇게 되면 하나의 버튼에 하나의 이미지를 가지고 표현을 할 수 있게 된다.

이와 마찬가지로, 다양한 종류의 이미지를 필요로 하는 모듈이더라도, 하나의 이미지에 맵(map)을 그리고, 그 맵(Map)에 따라 각 이미지를 배치한 후, 코드에서 잘라내어 쓰면 한 개의 이미지를 가지고 사용할 수 있게 된다. 본 발명에서는 2개의 이미지를 사용했지만, 1개의 이미지 파일만으로도 구현할 수 있다.

도 2는 본 발명의 실시에 따라 구현된 어플리케이션 프로그램을 나타내는 예시도이다.

어플리케이션 프로그램은 모두 3가지의 서브 모듈(210,220,230)로 구성되어 있다.

이러한 구성은 별도의 XML문서에서 정의하고 있고, 상기 XML문서에는 여러가지 상황에 따른 모드들을 포함한다. 각각의 상황에 맞는 모드를 상기 어플리케이션 프로그램에서 로딩(loading)하게 된다.

도 3은 도 2에서 도시한 어플리케이션 프로그램의 구성에 대한 설명을 하고 있는 XML문서를 나타내는 예시도로서, 음악을 청취할 수 있는 레이아웃(layout) 컴포넌트로 구성된 것을 나타내고 있다.

6라인에서는 도 2에서 도시한 어플리케이션 프로그램이 모두 3개의 컴포넌트를 갖고 있다는 것을 나타내고 있다.

7라인 내지 10라인에서는 210 컴포넌트를, 11라인 내지 14라인에서는 220 컴포넌트를, 15라인 내지 18라인에서는 230 컴포넌트를 설명하고 있다.

즉, 도 2에서 도시한 어플리케이션 프로그램에는 크게 3가지 서브 모듈이 있는데, 각각의 서브 모듈은 AVS2Title.dll, MusicStationContainer.dll, AVS2Bottom.dll에 정의가 되어 있다. 상기 어플리케이션 프로그램은 모드에 따라 하위에 어떤 모듈을 로딩(loading)할 것인지를 XML문서를 통해 알게 되는 것이다.

상기 220 컴포넌트에 해당하는 MusicStationContainer.dll에는 220 컴포넌트 자신의 구성에 대한 설명이 있는데, 이를 나타낸 것이 도 4이다. 도 4에서 보면 'PlayerMenuContainer', 'Toolbar4Player', 'MediaFrameContainer'의 이름을 갖는 3가지의 모듈이 있는 것을 알 수 있다. 즉, 도 3에서 나타난 'MusicStationContainer'라는 이름을 갖는 모듈은 3개의 컴포넌트를 갖는 윈테이너이라고도 할 수 있다. 또한 도 4에서 보면, MusicStationContainer에는 'Default', 'Simple', 'Remocon'의 3가지 모드가 있고, 각 모드에 대한 정의가 나타나 있다.

도 2내지 도 4에서 예시한 바와 같이, 프로그램 모듈의 구성을 나타내는 XML문서에 어떤 정보를 기술하느냐에 따라 어플리케이션 프로그램의 구성이 달라질 수 있는 것이다.

도 5는 본 발명의 실시에 따른 어플리케이션 프로그램에서 하나의 모듈을 로딩(loading)하는 방법을 나타내는 일실시예 처리 흐름도이다.

어플리케이션 프로그램이 시작하면 상기 어플리케이션 프로그램의 구성을 기술하고 있는 XML 문서가 위치한 경로를 얻은 후, 상기 경로에 위치한 XML 문서를 오픈(open)한다(S510). 그리고 나서, XML 문서를 파싱하여(S520) 필요한 정보를 추출한 후, 상기 정보를 이용하여 해당 DLL 리스트 및 이미지 리소스를 로딩(loading)하게 된다(S530). 마지막으로 상기 DLL 리스트에 기술된 모듈을 실행시키고 필요한 이미지를 불러 오게 됨으로써 화면을 구성하게 된다(S540).

도 6은 본 발명의 실시에 따른 각각의 모듈간의 데이터를 공유하는 방법을 나타내는 예시도이다.

본 발명에서 데이터의 공유가 필요한 이유는 하나의 모듈이 처리할 수 없는 일을 다른 모듈에게 명령을 전달할 필요가 있기 때문이다. 본 발명에 따른 어플리케이션 프로그램의 아키텍쳐에서는 자신 이외에 어떤 모듈이 있는지를 모른다. 각각의 모듈이 독립적이므로, 자신 이외에 어떤 모듈이 있는지 알 수 없다. 이것은 각 모듈간에 독립성을 나타내는 중요한 특성이다. 따라서, 각 모듈은 자신이 처리할 수 없는 일을 다른 모듈에게 알려, 처리할 수 있도록 해야 한다. 또한 각 모듈은 자신 이외에 다른 모듈이 메모리 상에 로딩(loading) 되었는지를 알 수 없다. 그러므로 기존의 함수를 호출하는 방식을 사용할

수가 없게 된다. 메모리 또는 파일과 같은 데이터 공유 저장소에 특정한 공간을 할당하여, 필요한 정보를 저장하고, 자신이 메모리 상에서 언로딩(unloading) 되더라도 정보가 남아있도록 하여, 다른 모듈에서 그 정보를 참작함으로써, 작업을 수행할 수 있도록 하는 방법이다.

예컨대, 미디어 파일을 관리하는 프로그램(이하, '라이브러리'라 한다)과, 미디어 파일을 재생하는 프로그램(이하, '플레이어'라고 한다)이 있다고 가정하면, 라이브러리와 플레이어는 본 발명에서는 각각의 별개의 모듈로 존재한다. 즉 각각의 모듈은 독립적이며, 자신 이외에 어떤 모듈이 있는지 모르고 있다. 이때, 라이브러리에서 하나의 파일을 실행하게 되면 (S610) 선택된 파일이 재생되어야 한다. 그러나, 이것은 라이브러리의 기능을 벗어나는 영역이다. 라이브러리에서는 재생할 수 없으므로, 라이브러리는 다른 모듈이 이 파일을 실행할 수 있도록 정보를 전달해야 한다. 이때 플레이어는 아직 메모리 상에 로딩(loading) 되지 않은 상태이므로, 직접 전달을 할 수는 없고, 공유 저장소에 실행할 파일의 정보를 기록하게 된다(S615,S620). 그리고 파일을 실행하라는 명령을 플레이어로 전달한다(S625). 파일을 실행하라는 명령은 그 명령 자체일 뿐, 그 안에 파일의 정보가 들어있지는 않다. 플레이어는 파일을 실행하라는 명령을 받으면 공유 저장소를 확인하여, 정보를 찾게 된다(S650). 이때 정보가 있으면 그 정보를 읽어와서 실행하고(S660), 없을 경우에는 기본 화면으로 나타나게 된다(S655).

이상에서 설명한 본 발명은, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 있어 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능하므로 전술한 실시예 및 첨부된 도면에 한정하는 것은 아니다.

발명의 효과

상기한 바와 같이 이루어진 본 발명에 따르면, 기능 변경이나, UI 변경이 자주 발생할 가능성이 있는 어플리케이션 프로그램을 효율적으로 개발할 수 있고, 새로운 기능의 추가, 삭제 등이 용이하며, XML 문서의 수정과 같이 어플리케이션 프로그램 외부에서 스킨(skin)을 변경할 수 있기 때문에 사용자는 간편하게 다양한 스킨(skin)을 이용할 수 있는 효과를 얻을 수 있다.

(57) 청구의 범위

청구항 1.

사용자 인터페이스 또는 그래픽 사용자 인터페이스 어플리케이션 프로그램에 있어서,

독립적으로 동작하고, 하위 모듈에 대한 정보가 XML 문서로 정의된 구조를 갖는 다수의 프로그램 모듈을 로딩하는 단계;

상기 로딩된 프로그램 모듈간에 데이터 통신을 하기 위한 공동의 정보 저장소를 생성하는 단계;

상기 모듈들 중에서 사용자로부터 명령을 입력받은 제1 모듈이 명령을 수행하기 위한 데이터를 상기 공동의 정보 저장소에 기록하는 단계; 및

상기 모듈들 중에서 상기 명령을 수행할 수 있는 제2 모듈이 상기 정보 저장소로부터 데이터를 읽은 후, 상기 데이터를 이용하여 상기 명령을 수행하는 단계를 포함하는 어플리케이션 프로그램 모듈의 동적 배치 방법.

청구항 2.

삭제

청구항 3.

제1항에 있어서,

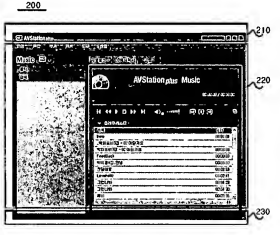
상기 프로그램 모듈은 사용자 인터페이스를 이용하여 사용자와 입출력 통신할 수 있는 컴포넌트 모듈과, 상기 컴포넌트 모듈을 다수 포함하는 컨테이너 모듈 및 상기 사용자 인터페이스를 제공하지 않고 사용자의 요구에 따른 동작을 수행하는 엔진 모듈을 포함하는 어플리케이션 프로그램 모듈의 동적 배치 방법.

도면

도면1

```
Line 1. <?xml version="1.0" encoding="euc-kr"?>
Line 2. <!DOCTYPE Layout SYSTEM "AVSLayout.dtd">
Line 3. <Layout type="fix" width="374" height="30">
Line 4. <name>DVDPlayerLayout</name>
Line 5. <dll_path></dll_path>
Line 6. <mode bkgnd="eaa.bmp"/>
Line 7. <compts compt_count="2">
Line 8.    <compt dll_path="dvdscreen.dll" compt_layout="commonscreen.xml">
Line 9.       <compt_name>dvdscreen</compt_name>
Line 10.      <compt_pos style="LEFT_FIXED TOP_FIXED" xPos="0" yPos="0" width="360" height="260"/>
Line 11.      <compt_skin skin_type="background" style="LEFT_FIXED TOP_FIXED"></compt_skin>
Line 12.   </compt>
Line 13.   <compt dll_path="dvdctrl.dll" compt_layout="horizontallayout.xml">
Line 14.      <compt_name>dvdcontrol</compt_name>
Line 15.      <compt_pos style="LEFT_FIXED BOTTOM_FIXED" xPos="0" yPos="264" width="380" height="30"/>
Line 16.      <compt_skin skin_type="background" style="LEFT_FIXED BOTTOM_FIXED HEIGHT_FIXED"></compt_skin>
Line 17.   </compt>
Line 18. </compts>
Line 19. </Layout>
```

도면2



도면3

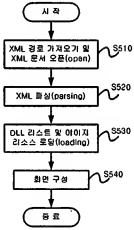
```
Line 1. <?xml version="1.0" encoding="euc-kr"?>
Line 2. <!DOCTYPE Layout SYSTEM "AVSLayout.dtd">
Line 3. <Layout type="resize" width="800" height="600" bkgnd="Common/Bk_Main.bmp" minX="272" minY="53">
Line 4. <name>AVStation Plus</name>
Line 5. <mode width="750" height="600">
Line 6.   <compts compt_count="3">
Line 7.     <compt dll_path="AVS2Title.dll">
Line 8.       <compt_name>AVS2Title.dll</compt_name>
Line 9.       <compt_pos style="LEFT_FIXED TOP_FIXED WIDTH_RATIO" xPos="0" yPos="36" width="750" height="543"
pWidth="100" pHeight="100"/>
Line 10.    </compt>
Line 11.  <compt dll_path="MusicStationContainer.dll">
Line 12.    <compt_name>MusicStationContainerAVS2Bottom</compt_name>
Line 13.    <compt_pos style="LEFT_FIXED TOP_FIXED WIDTH_RATIO HEIGHT_RATIO" xPos="0" yPos="36" width="750"
height="543" pWidth="100" pHeight="100"/>
Line 14.  </compt>
Line 15.    <compt dll_path="AVS2Bottom.dll">
Line 16.    <compt_name>AVS2Bottom</compt_name>
Line 17.    <compt_pos style="LEFT_FIXED BOTTOM_FIXED WIDTH_RATIO " xPos="0" yPos="0" width="750" height="21"
pWidth="100"/>
Line 18.    </compt>
Line 19.  </compts>
Line 20. </Layout>
```

도면4

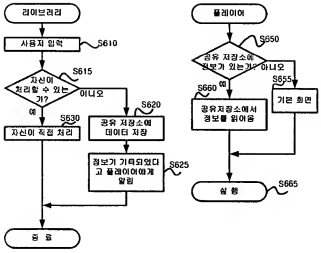
```
<?xml version="1.0" encoding="euc-kr"?>
<!DOCTYPE Layout SYSTEM "AVSLayout.dtd">
<Layout type="resize" width="749" height="545">
<name>AVS2_Main</name>
<dll_path/>
<mode bkgnd="MovieNMusicWAVS_Bk_XX_Station.bmp">
  <compts compt_count="3">
    <compt dll_path="PlayerMenuContainer.dll">
      <compt_name>PlayerMunuContainer.dll</compt_name>
      <compt_pos style="LEFT_FIXED TOP_FIXED WIDTH_RATIO" xPos="0" yPos="0" width="749" height="19"
pWidth="100" />
    </compt>
    <compt dll_path="Toolbar4Player.dll">
      <compt_name>Toolbar4Player</compt_name>
      <compt_pos style="LEFT_FIXED TOP_FIXED WIDTH_RATIO" xPos="0" yPos="19" width="749" height="50"
pWidth="100" />
    </compt>

    <compt dll_path="MediaFrameContainer.dll">
      <compt_name>MediaFrameContainer</compt_name>
      <compt_pos style="LEFT_FIXED TOP_FIXED WIDTH_RATIO HEIGHT_RATIO" xPos="15" yPos="69" width="719"
height="476" pWidth="100" pHeight="100"/>
    </compt>
  </compts>
</mode>
<mode>
.....
</mode>
<mode>
.....
</mode>
</Layout>
```

도면5



도면6


라이브러리
시용자 입력
S610
자신이 처리할 수 있는가?
S615
예
아니오
S630
자신이 직접 처리
S620
공유 저장소에 데이터 저장
정보가 기록되었다고 플레이어에게 알림
S625
종 료
플레이어
S650
공유 저장소에 정보가 있는가?
아니오
S655
기본 화면
예
S660
공유저장소에서 정보를 읽어옴
실 행
S665